Grid Technology: Enabling Scalable Virtual Organizations

Imran Khan and Dr. Nasir Tauheed
SZABIST
Karachi, Pakistan

Abstract

"Grid" computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation. In this report, we define this new field. First, we review the "Grid problem," which we define as flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources—what we refer to as virtual organizations. We describe requirements that we believe any such mechanisms must satisfy and we discuss the importance of defining a compact set of intergrid protocols to enable interoperability among different Grid systems. Finally, we discuss how Grid technologies relate to other contemporary technologies, including enterprise integration, application service provider, storage service provider, and peer-to-peer computing. We maintain that Grid concepts and technologies complement and have much to contribute to these other approaches.

1. INTRODUCTION

The term "the Grid" was coined in the mid1990s to denote a proposed distributed computing infrastructure for advanced science and engineering [1]. Considerable progress has since been made on the construction of such an infrastructure [2], but the term "Grid" has also been conflated, at least in popular perception, to embrace everything from advanced networking to artificial intelligence. One might wonder whether the term has any real substance and meaning. Is there really a distinct "Grid problem" and hence a need for new "Grid
technologies"? If so, what is the nature of these technologies, and what is their domain of applicability? While numerous groups have interest in Grid concepts and share, to a significant extent, a common vision of Grid architecture, we do not see consensus on the answers to these questions.

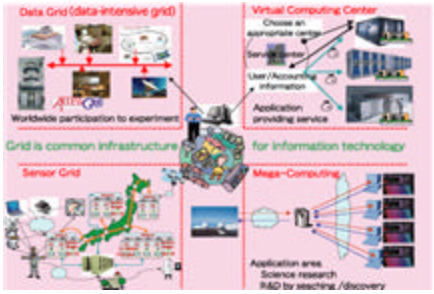

Figure 1: The Grid is common infrastructure for information technology [3]

What's Grid computing?

Sometimes it's easier to start defining Grid computing by telling you what it isn't. For instance, it's not artificial intelligence, and it's not some kind of advanced networking technology. It's also not some kind of science-fictional panacea to cure all of our technology ailments.

If you can think of the Internet as a network of communication, then Grid computing is a network of computation: tools and protocols for coordinated resource sharing and problem solving among pooled assets. These pooled assets are known as *virtual organizations*. They can be distributed across the globe; they're heterogeneous (some PCs, some servers, maybe mainframes and supercomputers); somewhat autonomous (a Grid can potentially access resources in different organizations); and temporary.

Figure 2: The Grid virtualizes herterogeneous and geographically disperse resource for each virtual organization presenting a simpler view [3]

Budgets are tight, resources are thin, and skilled human resources can be scarce or expensive. To top it off, most corporate managers know that they have a super-abundance of idle computing power. It's well known in industry circles that most desktop machines only use 5% to 10% of their capacity, and most servers barely peak out at 20%. No surprise then that many of the big money people in corporate America balk at the thought of purchasing more equipment to get the job done.[4]

What these companies need is not more horsepower, but more efficient use of existing horsepower. They need a way to tie all of these idle machines together into a pool of potential labor, manage those resources, and provide secure and reliable access to the number-crunching muscle.

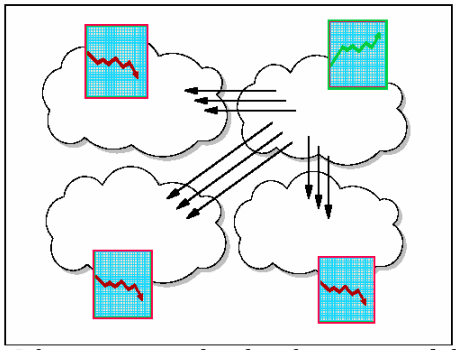

Figure 3: Jobs are migrated to less busy parts of the grid to balance resource loads and absorb unexpected peaks of activity in a part of an organization [3]

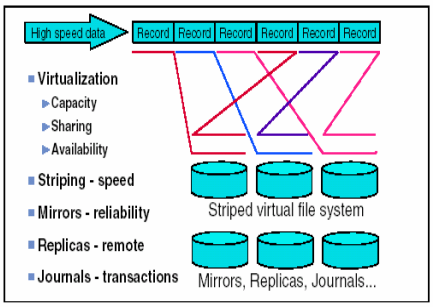

Figure 4: Data stripping is writing or reading successive records to/from different physical devices, overlapping the access for faster throughput additional technique increase reliability. **[3]**

Imagine if a corporation or organization could use all of its idle desktop PCs at night to run memory- and processor-intensive tasks? They would get more work done faster, possibly get to market faster, and at the same time cut down their IT expenses.

2. The Emergence of Virtual Organizations

Consider the following four scenarios:

- A company needing to reach a decision on the placement of a new factory invokes a sophisticated financial forecasting model from an ASP, providing it with access to appropriate proprietary historical data from a corporate database on storage systems operated by an SSP. [5] During the decision-making meeting, what-if scenarios are run collaboratively and interactively, even though the division heads participating in the decision are located in different cities.

- The ASP itself contracts with a cycle provider for additional "oomph" during particularly demanding scenarios, requiring of course that cycles meet desired security and performance requirements.

- An industrial consortium formed to develop a feasibility study for a next-generation supersonic aircraft undertakes a highly accurate multidisciplinary simulation of the entire aircraft. This simulation integrates proprietary software components developed by different participants, with each component operating on that participant's computers and having access to appropriate design databases and other data made available to the consortium by its members. [12]
- A crisis management team responds to a chemical spill by using local weather and soil models to estimate the spread of the spill, determining the impact based on population location as well as geographic features such as rivers and water supplies, creating a short-term mitigation plan (perhaps based on chemical reaction models), and tasking emergency response personnel by planning and coordinating evacuation, notifying hospitals, and so forth.
- Thousands of physicists at hundreds of laboratories and universities worldwide come together to design, create, operate, and analyze the products of a major detector at CERN, the European high energy physics laboratory. During the analysis phase, they pool their computing, storage, and networking resources to create a "Data Grid" capable of analyzing petabytes of data [7].

These four examples differ in many respects: the number and type of participants, the types of activities, the duration and scale of the interaction, and the resources being shared. But they also have much in common, as discussed in the following (see also Figure 5).

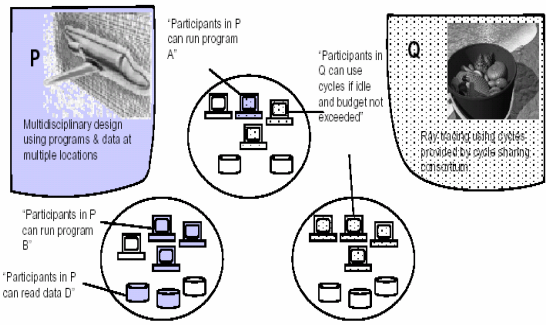

Figure 5: An actual organization can participate in one or more VOs by sharing some or all of its resources. [8]

In each case, a number of mutually distrustful participants with varying degrees of prior

relationship (perhaps none at all) want to share resources in order to perform some task. Furthermore, sharing is about more than simply document exchange (as in "virtual enterprises" [7]): it can involve direct access to remote software, computers, data, sensors, and other resources. For example, members of a consortium may provide access to specialized software and data and/or pool their computational resources.

Resource sharing is conditional: each resource owner makes resources available, subject to constraints on when, where, and what can be done. For example, a participant in VO P of Figure 5 might allow VO partners to invoke their simulation service only for "simple" problems. Resource consumers may also place constraints on properties of the resources they are prepared to work with. For example, a participant in VO Q might accept only pooled computational resources certified as "secure." [1] The implementation of such constraints requires mechanisms for expressing policies, for establishing the identity of a consumer or resource (authentication), and for determining whether an operation is consistent with applicable sharing relationships (authorization).

Sharing relationships can vary dynamically over time, in terms of the resources involved, the nature of the access permitted, and the participants to whom access is permitted. And these relationships do not necessarily involve an explicitly named set of individuals, but rather may be defined implicitly by the policies that govern access to resources. For example, an organization might enable access by anyone who can demonstrate that they are a "customer" or a "student."

The dynamic nature of sharing relationships means that we require mechanisms for discovering and characterizing the nature of the relationships that exist at a particular point in time. For example, a new participant joining VO Q must be able to determine what resources it is able to access, the "quality" of these resources, and the policies that govern access.

Sharing relationships are often not simply client-server, but peer to peer: providers can be consumers, and sharing relationships can exist among any subset of participants. Sharing relationships may be combined to coordinate use across many resources, each owned by different organizations. For example, in VO Q, a computation started on one pooled computational resource may subsequently access data or initiate sub computations elsewhere. The ability to delegate authority in controlled ways becomes important in such situations, as do mechanisms for coordinating operations across multiple resources (e.g., coscheduling).

The same resource may be used in different ways, depending on the restrictions placed on the sharing and the goal of the sharing. For example, a computer may be used only to run a specific piece of software in one sharing arrangement, while it may provide generic compute cycles in another. Because of the lack of a priori knowledge about how a resource may be used, performance metrics, expectations, and limitations (i.e., quality of service) may be part of the conditions placed on resource sharing or usage.

These characteristics and requirements define what we term a *virtual organization*, a concept that we believe is becoming fundamental to much of modern computing. VOs enable disparate groups of organizations and/or individuals to share resources in a controlled fashion, so that members may collaborate to achieve a shared goal.

3. The Nature of Grid Architecture

The establishment, management, and exploitation of dynamic, cross-organizational VO sharing relationships require new technology. We structure our discussion of this technology in terms of a *Grid architecture* that identifies fundamental system components, specifies the purpose and function of these components, and indicates how these components interact with one another.

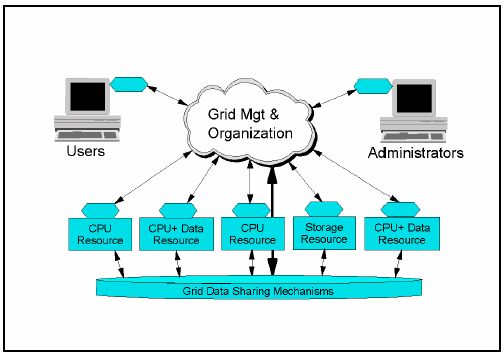

Figure 6: A Simple Grid[8]

In defining a Grid architecture, we start from the perspective that effective VO operation requires that we be able to establish sharing relationships among *any* potential participants [9]. Interoperability is thus the central issue to be addressed. In a networked environment, interoperability means common protocols. Hence, our Grid architecture is first and foremost a *protocol* architecture, with protocols defining the basic mechanisms by which VO users and resources negotiate, establish, manage, and exploit sharing relationships. A standards-based open architecture facilitates extensibility, interoperability, portability, and code sharing; standard protocols make it easy to define standard services that provide enhanced capabilities. We can also construct Application Programming Interfaces and Software Development Kits to provide the programming abstractions required to create a usable Grid.

4. Grid Architecture Description

Our goal in describing our Grid architecture is not to provide a complete enumeration of all required protocols (and services, APIs, and SDKs) but rather to identify requirements for general classes of component.

The result is an extensible, open architectural structure within which can be placed solutions to key VO requirements.

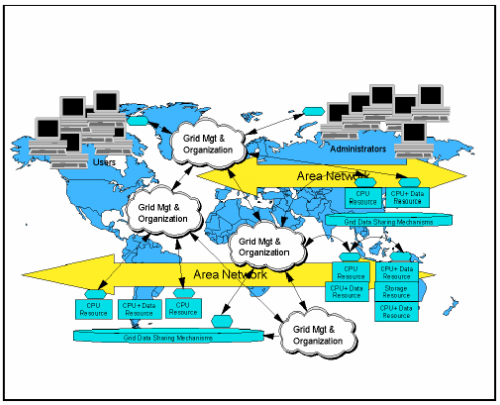

Figure 7: A more complex intergird[8]

Our architecture and the subsequent discussion organize components into layers, as shown in Figure 7. Components within each layer share common characteristics but can build on capabilities and behaviors provided by any lower layer. In specifying the various layers of the Grid architecture, we follow the principles of the "hourglass model" [2].

The narrow neck of the hourglass defines a small set of core abstractions and protocols (e.g., TCP and HTTP in the Internet), onto which many different high-level behaviors can be mapped (the top of the hourglass), and which themselves can be mapped onto many different underlying technologies (the base of the hourglass). By definition, the number of protocols defined at the neck must be small. In our architecture, the neck of the hourglass consists of *Resource* and *Connectivity* protocols, which facilitate the sharing of individual resources. Protocols at these layers are designed so that they can be implemented on top of a diverse range of resource types, defined at the *Fabric* layer, and can in turn be used to construct a wide range of global services and application-specific behaviors at the *Collective* layer—so called because they involve the coordinated ("collective") use of multiple resources.

Our architectural description is high level and places few constraints on design and implementation. To make this abstract discussion more concrete, we also list, for illustrative purposes, the protocols defined within the Globus Toolkit [12], and used within such Grid projects as the NSF's National Technology Grid, NASA's Information Power Grid [4], Particle Physics Data Grid (www.ppdg.net), and the European Data Grid (www.eu-datagrid.org).

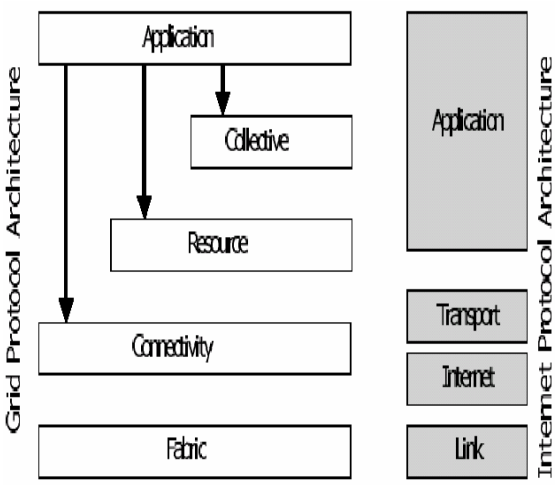

Figure 8: Grid Protocol Architecture & Internet Protocol Architecture [8]

Fabric: Interfaces to Local Control

The Grid *Fabric* layer provides the resources to which shared access is mediated by Grid protocols: for example, computational resources, storage systems, catalogs, network resources, and sensors. A "resource" may be a logical entity, such as a distributed file system, computer cluster, or distributed computer pool; in such cases, a resource implementation may involve internal protocols (e.g., the NFS storage access protocol or a cluster resource management system's process management protocol), but these are not the concern of Grid architecture.

Fabric components implement the local, resource-specific operations that occur on specific resources (whether physical or logical) as a result of sharing operations at higher levels. There is thus a tight and subtle interdependence between the functions implemented at the Fabric level, on the one hand, and the sharing operations supported, on the other. Richer Fabric functionality enables more sophisticated sharing operations; at the same time, if we place few demands on Fabric elements, then deployment of Grid infrastructure is simplified. For example, resourcelevel support for advance reservations makes it possible for higher-level services to aggregate (coschedule) resources in interesting ways that would otherwise be impossible to achieve.

The following brief and partial list provides a resourcespecific characterization of capabilities.

• ***Computational resources*:** Mechanisms are required for starting programs and for monitoring and controlling the execution of the resulting processes. Management mechanisms that allow control over the resources allocated to processes are useful, as are advance reservation mechanisms. Enquiry functions are needed for determining hardware and software characteristics as well as relevant state information

such as current load and queue state in the case of scheduler-managed resources.

• ***Storage resources*:** Mechanisms are required for putting and getting files. Third-party and high-performance (e.g., striped) transfers are useful. So are mechanisms for reading and writing subsets of a file and/or executing remote data selection or reduction functions [9]. Management mechanisms that allow control over the resources allocated to data transfers (space, disk bandwidth, network bandwidth, CPU) are useful, as are advance reservation mechanisms. Enquiry functions are needed for determining hardware and software characteristics as well as relevant load information such as available space and bandwidth utilization.

• ***Network resources*:** Management mechanisms that provide control over the resources allocated to network transfers (e.g., prioritization, reservation) can be useful. Enquiry functions should be provided to determine network characteristics and load.

• ***Code repositories*:** This specialized form of storage resource requires mechanisms for managing versioned source and object code: for example, a control system such as CVS.

• ***Catalogs*:** This specialized form of storage resource requires mechanisms for
implementing catalog query and update operations: for example, a relational database [8].

Connectivity:

The *Connectivity* layer defines core communication and authentication protocols required for Grid-specific network transactions. Communication protocols enable the exchange of data between Fabric layer resources. Authentication protocols build on communication services to provide cryptographically secure mechanisms for verifying the identity of users and resources.

With respect to security aspects of the Connectivity layer, we observe that the complexity of the security problem makes it important that any solutions be based on existing standards whenever possible. As with communication, many of the security standards developed within the context of the Internet protocol suite are applicable.

Authentication solutions for VO environments should have the following characteristics [5]:

• ***Single sign on.*** Users must be able to "log on" (authenticate) just once and then have access to multiple Grid resources defined in the Fabric layer, without further user intervention.

• ***Delegation*** [4]. A user must be able to endow a program with the ability to run
on that user's behalf, so that the program is able to access the resources on which the user is authorized. The program should (optionally) also be able to conditionally delegate a subset of its rights to another program (sometimes referred to as restricted delegation).

• ***Integration with various local security solutions*:** Each site or resource provider may employ any of a variety of local security solutions, including Kerberos and Unix security. Grid security solutions must be able to interoperate with these various local solutions. They cannot, realistically, require wholesale replacement of local security solutions but rather must allow mapping into the local environment.

• ***User-based trust relationships*:** In order for a user to use resources from multiple providers together, the security system must not require each of the resource providers to cooperate or interact with each other in configuring the security environment. For example, if a user has the right to use sites A and B, the user should be able to use sites A and B together without requiring that A's and B's security administrators interact.

Grid security solutions should also provide flexible support for communication protection (e.g., control over the degree of protection, independent data unit protection for unreliable protocols, support for reliable transport protocols other than TCP) and enable stakeholder control over authorization decisions, including the ability to restrict the delegation of rights in various ways.

Resource: Sharing Single Resources

The Resource layer builds on Connectivity layer communication and authentication protocols to define protocols (and APIs and SDKs) for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources. Resource layer implementations of these protocols call Fabric layer functions to access and control local resources. Resource layer protocols are concerned entirely with individual resources and hence ignore issues of global state and atomic actions across distributed collections; such issues are the concern of the Collective layer discussed next.
Two primary classes of Resource layer protocols can be distinguished:

• ***Information protocols*** are used to obtain information about the structure and state of a resource, for example, its configuration, current load, and usage policy (e.g., cost).

• ***Management protocols*** are used to negotiate access to a shared resource, specifying, for example, resource requirements (including advanced reservation and quality of

service) and the operation(s) to be performed, such as process creation, or data access. Since management protocols are responsible for instantiating sharing relationships, they must serve as a "policy application point," ensuring that the requested protocol operations are consistent with the policy under which the resource is to be shared.

***Globus Toolkit*:** A small and mostly standards-based set of protocols is adopted. In particular:

• A **Grid Resource Information Protocol** (**GRIP**, currently based on the Lightweight Directory Access Protocol: LDAP) is used to define a standard resource information protocol and associated information model. An associated soft-state resource registration protocol, the Grid Resource Registration Protocol (GRRP), is used to register resources with Grid Index Information Servers, discussed in the next section [12].

• The HTTP-based **Grid Resource Access and Management (GRAM)** protocol is used for allocation of computational resources and for monitoring and control of computation on those resources.

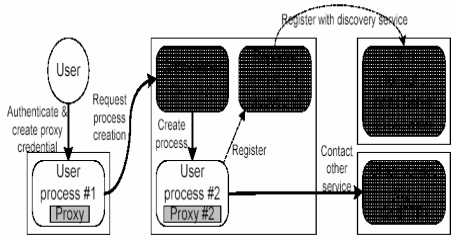

Figure 9: Selected Globus Toolkit mechanisms [4]

• An extended version of the File Transfer Protocol, **GridFTP**, is a management protocol for data access; extensions include use of Connectivity layer security protocols, partial file access, and management of parallelism for high-speed transfers [1]. FTP is adopted as a base data transfer protocol because of its support for third-party transfers and because its separate control and data channels facilitate the implementation of sophisticated servers.

• **LDAP** is also used as a **catalog access protocol**.

Collective: Coordinating Multiple Resources

While the Resource layer is focused on interactions with a single resource, the next layer in the architecture contains protocols and services (and APIs and SDKs) that are not associated with any one specific resource but rather are global in nature and capture interactions across collections of resources. For example:

• ***Directory services*** allow VO participants to discover the existence and/or properties of VO resources. A directory service may allow its users to query for resources by name and/or by attributes such as type, availability, or load [12]. Resource-level GRRP and GRIP protocols are used to construct directories.

• ***Co-allocation, scheduling, and brokering services*** allow VO participants to request the allocation of one or more resources for a specific purpose and the scheduling of tasks on the appropriate resources

• ***Monitoring and diagnostics services*** support the monitoring of VO resources for failure, adversarial attack ("intrusion detection"), overload, and so forth.

• ***Data replication services*** support the management of VO storage (and perhaps also network and computing) resources to maximize data access performance with respect to metrics such as response time, reliability, and cost [1].

• ***Grid-enabled programming*** *systems* enable familiar programming models to be used in Grid environments, using various Grid services to address resource discovery, security, resource allocation, and other concerns. Examples include Grid-enabled implementations of the Message Passing Interface [1] and manager-worker frameworks [9].

• ***Workload management systems and collaboration frameworks***—also known as problem solving environments ("PSEs")—provide for the description, use, and management of multi-step, asynchronous, multi-component workflows

• ***Software discovery services*** discover and select the best software implementation and execution platform based on the parameters of the problem being solved [4].

• ***Community authorization servers*** enforce community policies governing resource access, generating capabilities that community members can use to access community resources.

These servers provide a global policy enforcement service by building on resource information, and resource management protocols (in the Resource layer) and security protocols in the Connectivity layer. Akenti [12] addresses some of these issues.

• ***Community accounting and payment services*** gather resource usage information for the purpose of accounting, payment, and/or limiting of resource usage by community members.

• ***Collaboratory services*** support the coordinated exchange of information within potentially large user communities, whether synchronously or asynchronously.

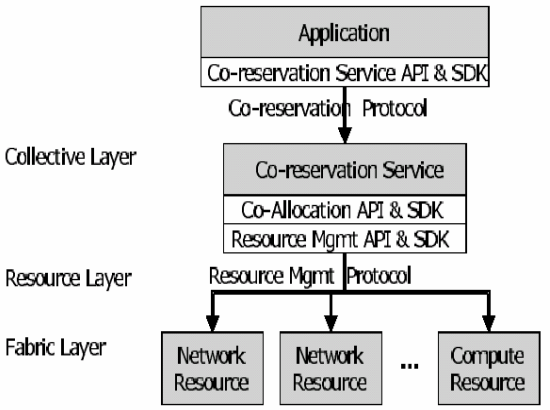

Figure 10: Collective and Resource layer protocols, services, APIs, and SDKS can be combined in a variety of ways to deliver functionality to applications. [4]

5. Applications

The final layer in our Grid architecture comprises the user applications that operate within a VO environment. Figure 13 illustrates an application programmer's view of Grid architecture.

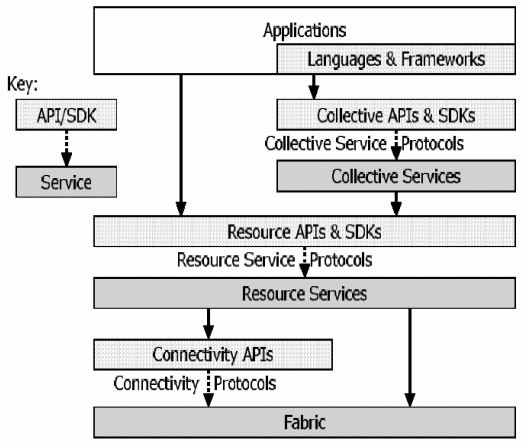

Figure 11: APIs are implemented by software development kits (SDKs), which in turn use Grid protocols to interact with network services that provide capabilities to the end user. [9]

6. Relationships with Other Technologies

The concept of controlled, dynamic sharing within VOs is so fundamental that we might assume that Grid-like technologies must surely already be widely deployed. In practice, however, while the need for these technologies is indeed widespread, in a wide variety of different areas we find only primitive and inadequate solutions to VO problems. In brief, current distributed computing approaches do not provide a general resource-sharing framework that addresses VO requirements.

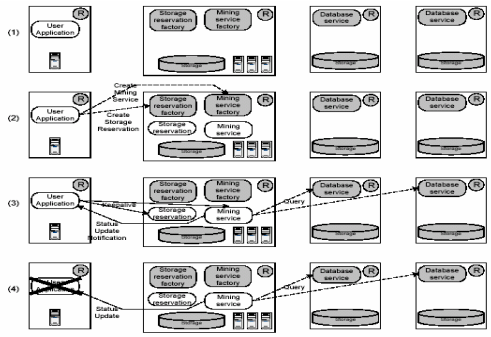

Figure 11: An example of Grid services at work. [1]

World Wide Web

The ubiquity of Web technologies (i.e., IETF and W3C standard protocols—TCP/IP, HTTP, SOAP, etc.—and languages, such as HTML and XML) makes them attractive as a platform for constructing VO systems and applications. However, while these technologies do an excellent job of supporting the browser-client-to-web-server interactions that are the foundation of today's Web, they lack features required for the richer interaction models that occur in VOs. [7]For example, today's Web browsers typically use TLS for authentication, but do not support single sign-on or delegation.

Application and Storage Service Providers

Application service providers, storage service providers, and similar hosting companies typically offer to outsource specific business and engineering applications (in the case of ASPs) and storage capabilities (in the case of SSPs).[9]

7. Other Perspectives on Grids

The perspective on Grids and VOs presented in this article is of course not the only view that can be taken. We summarize here—and critique—some alternative perspectives (given in italics). [9]

The Grid is a next-generation Internet.
"The Grid" is not an alternative to "the Internet": it is rather a set of additional protocols and services that build on Internet protocols and services to support the creation and use of computation- and data-enriched environments. Any resource that is "on the Grid" is also, by definition, "on the Net."

The Grid is a source of free cycles.
Grid computing does not imply unrestricted access to resources. Grid computing is about controlled sharing. Resource owners will typically want to enforce policies that constrain access according to group membership, ability to pay, and so forth. Hence, accounting is important, and a Grid architecture must incorporate resource and collective protocols for exchanging usage and cost information, as well

as for exploiting this information when deciding whether to enable sharing.

The Grid requires a distributed operating system.
In this view, Grid software should define the operating system services to be installed on every participating system, with these services providing for the Grid what an operating system provides for a single computer: namely, transparency with respect to location, naming, security, and so forth. Put another way, this perspective views the role of Grid software as defining a virtual machine. The tremendous physical and administrative heterogeneities encountered in Grid environments means that the traditional transparencies are unobtainable; on the other hand, it does appear feasible to obtain agreement on standard protocols.

The Grid requires new programming models.
Programming in Grid environments introduces challenges that are not encountered in sequential (or parallel) computers, such as multiple administrative domains, new failure modes, and large variations in performance.

The Grid makes high-performance computers superfluous.
The hundreds, thousands, or even millions of processors that may be accessible within a VO represent a significant source of computational power, if they can be harnessed in a useful fashion. Grid computing may well increase, rather than reduce, demand for such systems by making access easier.

8. The Present And The Future
Today, grid systems are still at the early stages of providing a reliable, well performing, and automatically recoverable virtual data sharing and storage. We will see products that take on this task in a grid setting, federating data of all kinds, and achieving better performance, integration with scheduling, reliability, and capacity. Autonomic computing has the goal to make the administrator's job easier by automating the various complicated tasks involved in managing a grid. It will make it easier to assemble the best products from various vendors, increasing the overall value of grid computing.

9. What The Grid Cannot Do

A word of caution should be given to the overly enthusiastic. The grid is not a silver bullet that can take any application and run it a 1000 times faster without the need for buying any more machines or software. Not every application is suitable or enabled for running on a grid. [4]

Some kinds of applications simply cannot be parallelized. For others, it can take a large amount of work to modify them to achieve faster throughput. The configuration of a grid can greatly affect the performance, reliability, and security of an organization's computing infrastructure. For all of these reasons, it is important for the us to understand how far the grid has evolved today and which features are coming tomorrow or in the distant future.

10. Summary

We have provided in this article a concise statement of the "Grid problem," which we define as controlled and coordinated resource sharing and resource use in dynamic, scalable virtual organizations. We have also presented both requirements and a framework for a Grid architecture, identifying the principal functions required to enable sharing within VOs and defining key relationships among these different functions. Finally, we have discussed in some detail how Grid technologies relate to other important technologies.

We hope that the vocabulary and structure introduced in this document will prove useful to the emerging Grid community, by improving understanding of our problem and providing a common language for describing solutions.

The discussion in this study also raises a number of important questions. What are appropriate choices for the Intergrid protocols that will enable interoperability among Grid systems? What services should be present in a persistent fashion (rather than being duplicated by each application) to create usable Grids? And what are the key APIs and SDKs that must be delivered to users in order to accelerate development and deployment of Grid applications? We have our own opinions on these questions, but the answers clearly require further research.

REFERENCES

[1] Berman, F., Wolski, R., Figueira, S., Schopf, J. and Shao, G. Application- Level Scheduling on Distributed Heterogeneous Networks. In *Proc. Supercomputing '96*, 1996.

[2] Beiriger, J., Johnson, W., Bivens, H., Humphreys, S. and Rhea, R., Constructing the ASCI Grid. In *Proc. 9th IEEE Symposium on High Performance Distributed Computing*, 2000, IEEE Press.

[3] www.ibm.com/redbooks

[4] Berman, F. High-Performance Schedulers. In Foster, I. and Kesselman, C. eds. *The Grid:Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 279-309.

[5] Benger, W., Foster, I., Novotny, J., Seidel, E., Shalf, J., Smith, W. and alker, P., Numerical Relativity in a Distributed Environment. In *Proc. 9th SIAM Conference on Parallel Processing for Scientific Computing*, 1999.

[6] Aiken, R., Carey, M., Carpenter, B., Foster, I., Lynch, C., Mambretti, J., Moore, R.,Strasnner, J. and Teitelbaum, B. Network Policy and Services: A Report of a Workshop on Middleware, IETF, RFC 2768, 2000. http://www.ietf.org/rfc/rfc2768.txt.

[7] Barry, J., Aparicio,M: An Investigation of Distributed Object Approaches to Support MES Development and Deployment in a Virtual Enterprise. In *2nd Intl Enterprise Distributed Computing Workshop*, 1998, IEEE Press.

[8] www.globus.org/research/papers/anatomy.pdf

[9] Dierks, T. and Allen, C. The TLS Protocol Version 1.0, IETF, RFC 2246, 1999. http://www.ietf.org/rfc/rfc2246.txt.

[10] *Realizing the Information Future: The Internet and Beyond*. National Academy Press,1994. http://www.nap.edu/readingroom/books/rtif/.

[11] Baker, F. Requirements for IP Version 4 Routers, IETF, RFC 1812, 1995. http://www.ietf.org/rfc/rfc1812.txt

[12] Dinda, P. and O'Hallaron, D., An Evaluation of Linear Models for Host Load Prediction. In *Proc. 8th IEEE Symposium on High-Performance Distributed Computing*, 1999, IEEE Press.

[13] Allcock, B., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S., Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. In *Mass Storage Conference*, 2001.

[14] Abramson, D., Sosic, R., Giddy, J. and Hall, B. Nimrod: A Tool for Performing Parameterized Simulations Using Distributed Workstations. In *Proc. 4th IEEE Symp. On High Performance Distributed Computing*, 1995.

[15] Arnold, K., O'Sullivan, B., Scheifler, R.W., Waldo, J. and Wollrath, A. *The Jini Specification*. Addison-Wesley, 1999. See also www.sun.com/jini.